**DISASTER MANAGEMENT SYSTEM**

**A MINI-PROJECT BY:**

**SIVA BALAN K 230701316**

**SANJAI KUMARAN K 230701284**

***in partial fulfilment of the award of the degree***

***OF***

***BACHELOR OF ENGINEERING***

**IN**

# COMPUTER SCIENCE AND ENGINEERING



**RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI**

**An Autonomous Institute**

**CHENNAI**

**NOVEMBER 2024**

# BONA FIDE CERTIFICATE

Certified that this project **“DISASTER MANAGEMENT SYSTEM”** is the bona fide work of **“SIVA BALAN K, SANJAI KUMARAN K ”** who carried out the project work under my supervision.

Submitted for the practical examination held on

|  |  |
| --- | --- |
| SIGNATURE  Mr. G SARAVANA GOKUL  Assistant Professor (SS),  Computer Science and Engineering,  Rajalakshmi Engineering College  (Autonomous),  Thandalam, Chennai-602105 | SIGNATURE  Ms. V. JANANEE  Assistant Professor (SG),  Computer Science and Engineering,  Rajalakshmi Engineering College  (Autonomous),  Thandalam, Chennai-602105 |
| **INTERNAL EXAMINER** | **EXTERNAL EXAMINER** |

## ABSTRACT

A **Disaster Management System** is a tool designed to help communities and organizations effectively prepare for, respond to, and recover from disasters. It integrates various technologies to monitor potential risks, provide early warnings, and coordinate emergency responses. The system helps in organizing resources like medical supplies, food, and rescue teams to ensure timely support during emergencies. By using mapping tools, it identifies high-risk areas and optimizes relief efforts. It also stores data for analyzing past disasters, improving future preparedness. Overall, the system aims to reduce the impact of disasters, protect lives, and assist in rebuilding affected communities.

A Disaster Management System is a comprehensive framework that helps in managing and reducing the effects of disasters, whether natural or man-made. It focuses on early detection and warning, allowing authorities to respond quickly and efficiently. The system coordinates efforts between various agencies, ensuring the proper distribution of resources like rescue teams, medical aid, and supplies to affected areas. By using technology such as real-time monitoring and mapping, it identifies risk zones and guides emergency responses. The system also plays a vital role in recovery and rebuilding efforts, helping communities return to normalcy while improving preparedness for future disasters.

A Disaster Management System is a platform designed to minimize the impact of disasters by enabling efficient preparation, response, and recovery. It helps monitor risks, issue early warnings, and coordinate emergency efforts to protect lives and property. By bringing together technology and resources, the system ensures timely deployment of aid and rescue operations to affected areas. It also supports rebuilding efforts and collects valuable data to improve future disaster preparedness and resilience, ultimately helping communities recover faster and more effectively.

## TABLE OF CONTENTS

### 1. INTRODUCTION

1.1. INTRODUCTION

1.2. IMPLEMENTATION

1.3. SCOPE OF THE PROJECT

### 2. SYSTEM SPECIFICATION

2.1. HARDWARE SPECIFICATION

2.2. SOFTWARE SPECIFICATION

1. **ENTITY RELATION MODEL**

3.1. ER DIAGRAM

### 4. SAMPLE CODE

4.1. LOGIN PAGE

4.2. DASHBOARD

4.3. OWNER PAGE

### 5. SNAPSHOTS

5.1. LOGIN PAGE

5.2. DASHBOARD

5.2 ADD DISASTER REPORT

5.3 VIEW DISASTER REPORT

5.4REMOVE DISASTER REPORT

**6. CONCLUSION**

**7. REFERENCES**

## INTRODUCTION

### 1.1 INTRODUCTION

Disaster management systems are vital tools for effectively addressing the challenges posed by natural and human-made disasters. These systems are designed to reduce the impact of disasters by improving preparedness, enabling efficient responses, and supporting recovery efforts. By integrating advanced technologies such as real-time monitoring, geographic information systems (GIS), and communication networks, disaster management systems help in early detection, risk assessment, and resource allocation.

1.2 IMPLEMENTATION

The Disaster management systems project is implemented using Java Swing for the user interface and MySQL for database management.

### 1.3 SCOPE OF THE PROJECT

The scope of a **Disaster Management System** includes developing a platform to monitor risks, provide early warnings, and coordinate disaster responses. It focuses on resource allocation, public awareness, and real-time communication to minimize the impact of disasters. The system also supports data analysis for risk assessment and future preparedness, ensuring efficient recovery and resilience in affected communities.

## SYSTEM SPECIFICATIONS

**2.1 HARDWARE SPECIFICATIONS:**

PROCESSOR: Intel i5

MEMORY SIZE: 16GB

HARD DISK: 500 GB of free space

**2.2 SOFTWARE SPECIFICATIONS:**

PROGRAMMING LANGUAGE: Java, SQL

FRONT-END: Java Swing

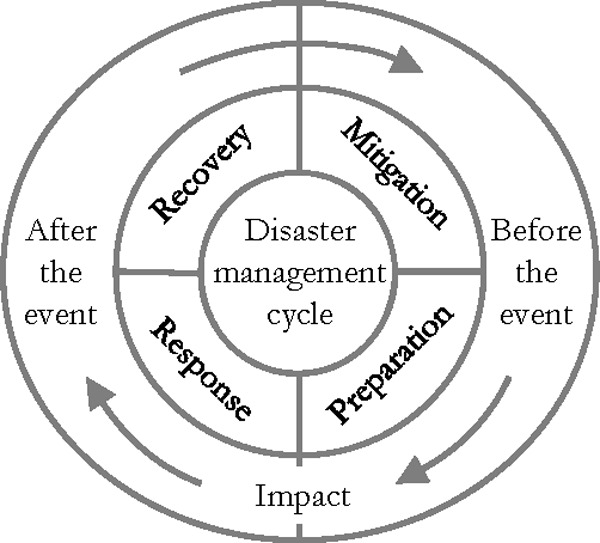
BACK-END: MySQL

OPERATING SYSTEM: Windows 11

## ENTITY RELATION MODEL

### 3.1 ER DIAGRAM

An Entity Relationship (ER) diagram outlines the structure of the database by representing its entities and their relationships. In this **Property Management System**, the Admin can log in to manage key operations, such as adding property details, viewing all property records, and deleting outdated or irrelevant data. In this simplified model, there are no complex relationships between the Property entity and other entities. All essential information, such as property type, location, owner details, and rental status, is directly associated with the property record.



## SOURCE CODE

**4.1 LOGIN PAGE:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class Loginpage extends JFrame {

private JTextField usernameField;

private JPasswordField passwordField;

private JButton loginButton;

private JButton registerButton;

private JLabel messageLabel;

public Loginpage() {

// Frame settings

setTitle("Login Page");

setExtendedState(JFrame.MAXIMIZED\_BOTH); // Set window to full screen

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create main panel with white background

JPanel mainPanel = new JPanel() {

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

// Load background image

ImageIcon icon = new ImageIcon("flood\_background.jpg"); // Update with your image path

Image image = icon.getImage();

// Scale the image to fit the entire window

g.drawImage(image, 0, 0, getWidth(), getHeight(), this);

}

};

mainPanel.setLayout(new BorderLayout(10, 10));

mainPanel.setBorder(BorderFactory.createEmptyBorder(50, 50, 50, 50)); // Increased padding

// Create form panel

JPanel formPanel = new JPanel(new GridBagLayout());

formPanel.setBackground(new Color(255, 255, 255, 150)); // Semi-transparent background

GridBagConstraints gbc = new GridBagConstraints();

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.insets = new Insets(10, 10, 10, 10); // Increased spacing

// UI Components with styled fonts and colors - increased sizes for better visibility

Font labelFont = new Font("Arial", Font.BOLD, 18); // Increased font size

Font fieldFont = new Font("Arial", Font.PLAIN, 18); // Increased font size

JLabel titleLabel = new JLabel("User Login", SwingConstants.CENTER);

titleLabel.setFont(new Font("Arial", Font.BOLD, 36)); // Increased title size

titleLabel.setForeground(new Color(51, 51, 51));

JLabel usernameLabel = new JLabel("Username:");

usernameLabel.setFont(labelFont);

usernameField = new JTextField(20);

usernameField.setFont(fieldFont);

usernameField.setPreferredSize(new Dimension(300, 40)); // Increased field size

JLabel passwordLabel = new JLabel("Password:");

passwordLabel.setFont(labelFont);

passwordField = new JPasswordField(20);

passwordField.setFont(fieldFont);

passwordField.setPreferredSize(new Dimension(300, 40)); // Increased field size

loginButton = new JButton("Login");

registerButton = new JButton("Register");

messageLabel = new JLabel("", SwingConstants.CENTER);

messageLabel.setForeground(Color.RED);

messageLabel.setFont(new Font("Arial", Font.BOLD, 16));

// Style buttons - both in blue with increased size

Color buttonBlue = new Color(0, 120, 215);

Dimension buttonSize = new Dimension(150, 50); // Increased button size

loginButton.setBackground(buttonBlue);

loginButton.setForeground(Color.BLACK);

loginButton.setFocusPainted(false);

loginButton.setFont(new Font("Arial", Font.BOLD, 18));

loginButton.setPreferredSize(buttonSize);

registerButton.setBackground(buttonBlue);

registerButton.setForeground(Color.BLACK);

registerButton.setFocusPainted(false);

registerButton.setFont(new Font("Arial", Font.BOLD, 18));

registerButton.setPreferredSize(buttonSize);

// Add components to form panel with increased spacing

gbc.gridx = 0;

gbc.gridy = 0;

gbc.gridwidth = 2;

gbc.insets = new Insets(0, 0, 50, 0); // Extra space below title

formPanel.add(titleLabel, gbc);

gbc.insets = new Insets(10, 10, 10, 10);

gbc.gridy = 1;

gbc.gridwidth = 1;

formPanel.add(usernameLabel, gbc);

gbc.gridx = 1;

formPanel.add(usernameField, gbc);

gbc.gridx = 0;

gbc.gridy = 2;

formPanel.add(passwordLabel, gbc);

gbc.gridx = 1;

formPanel.add(passwordField, gbc);

// Button panel with increased spacing

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 3));

buttonPanel.setBackground(Color.WHITE);

buttonPanel.add(loginButton);

buttonPanel.add(registerButton);

gbc.gridx = 0;

gbc.gridy = 3;

gbc.gridwidth = 2;

gbc.insets = new Insets(30, 0, 20, 0); // Extra space above buttons

formPanel.add(buttonPanel, gbc);

gbc.gridy = 4;

gbc.insets = new Insets(20, 0, 0, 0);

formPanel.add(messageLabel, gbc);

// Center the form in the main panel

JPanel centeringPanel = new JPanel(new GridBagLayout());

centeringPanel.setBackground(Color.WHITE);

centeringPanel.add(formPanel);

// Add centering panel to main panel

mainPanel.add(centeringPanel, BorderLayout.CENTER);

// Add main panel to frame

add(mainPanel);

// Action listener for the login button

loginButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String username = usernameField.getText();

String password = new String(passwordField.getPassword());

if (validateLogin(username, password)) {

messageLabel.setForeground(new Color(60, 179, 113));

messageLabel.setText("Login successful!");

dispose(); // Close the login page window

new Dashboard().setVisible(true); // Open the Dashboard page

} else {

messageLabel.setForeground(Color.RED);

messageLabel.setText("Invalid credentials.");

}

}

});

// Action listener for the register button

registerButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new Register(); // Open Register page

}

});

}

// Method to validate login credentials against the database

private boolean validateLogin(String username, String password) {

String url = "jdbc:mysql://localhost:3306/unisoft?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";

String user = "root";

String pass = "siva2005";

try (Connection con = DriverManager.getConnection(url, user, pass)) {

String query = "SELECT \* FROM users WHERE username = ? AND password = ?";

PreparedStatement pst = con.prepareStatement(query);

pst.setString(1, username);

pst.setString(2, password);

ResultSet rs = pst.executeQuery();

return rs.next(); // Returns true if the username and password are found in the database

} catch (SQLException e) {

e.printStackTrace();

messageLabel.setText("Database error.");

return false;

}

}

// Main method to launch the login page

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

try {

// Set system look and feel

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (Exception e) {

e.printStackTrace();

}

Loginpage loginPage = new Loginpage();

loginPage.setVisible(true); // Show login page

}

     });

    }

}

**4.2 DASHBOARD DESIGN:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class Dashboard extends JFrame {

private JButton addReportButton;

private JButton viewReportsButton;

private JButton removeReportsButton;

private JButton logoutButton;

private ImageIcon backgroundImage; // Store the image as ImageIcon

public Dashboard() {

// Load the flood background image

backgroundImage = new ImageIcon("flood\_background.jpg"); // Replace with your image path

// Setting up the frame

setTitle("Disaster Management Dashboard");

setExtendedState(JFrame.MAXIMIZED\_BOTH); // Set window to full screen

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create main panel with image background

JPanel mainPanel = new JPanel() {

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

// Draw the background image scaled to the panel's size

if (backgroundImage != null) {

Image img = backgroundImage.getImage(); // Convert to Image

g.drawImage(img, 0, 0, getWidth(), getHeight(), this); // Scale the image to fit the screen

}

}

};

mainPanel.setLayout(new BorderLayout());

// Header Panel

JPanel headerPanel = new JPanel(new BorderLayout());

headerPanel.setOpaque(false);

headerPanel.setBorder(BorderFactory.createEmptyBorder(30, 50, 30, 50));

JLabel titleLabel = new JLabel("Disaster Management System", SwingConstants.CENTER);

titleLabel.setFont(new Font("Arial", Font.BOLD, 40));

titleLabel.setForeground(Color.black); // Adjust to ensure visibility on the background image

headerPanel.add(titleLabel, BorderLayout.CENTER);

// Buttons Panel

JPanel buttonsPanel = new JPanel(new GridBagLayout());

buttonsPanel.setOpaque(false);

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(20, 20, 20, 20);

// Create and style buttons

Dimension buttonSize = new Dimension(300, 80);

Font buttonFont = new Font("Arial", Font.BOLD, 20);

Color buttonBlue = new Color(0, 120, 215);

addReportButton = createStyledButton("Add Disaster Report", buttonSize, buttonFont, buttonBlue);

viewReportsButton = createStyledButton("View Disaster Reports", buttonSize, buttonFont, buttonBlue);

removeReportsButton = createStyledButton("Remove Disaster Reports", buttonSize, buttonFont, buttonBlue);

logoutButton = createStyledButton("Logout", new Dimension(200, 60), buttonFont, new Color(220, 53, 69));

// Add buttons to panel

gbc.gridx = 0;

gbc.gridy = 0;

buttonsPanel.add(addReportButton, gbc);

gbc.gridy = 1;

buttonsPanel.add(viewReportsButton, gbc);

gbc.gridy = 2;

buttonsPanel.add(removeReportsButton, gbc);

gbc.gridy = 3;

gbc.insets = new Insets(50, 20, 20, 20); // Extra space above logout

buttonsPanel.add(logoutButton, gbc);

// Add panels to main panel

mainPanel.add(headerPanel, BorderLayout.NORTH);

mainPanel.add(buttonsPanel, BorderLayout.CENTER);

// Add main panel to frame

add(mainPanel);

// Action listeners

addReportButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new AddReportPage().setVisible(true); // Placeholder class for adding reports

}

});

viewReportsButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new ViewDisasterPage().setVisible(true); // Placeholder class for viewing disaster reports

}

});

removeReportsButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new RemoveDisasterPage().setVisible(true); // Placeholder class for removing disaster reports

}

});

logoutButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

dispose();

new Loginpage().setVisible(true); // Placeholder class for the login page

}

});

// Add hover effects to buttons

addHoverEffect(addReportButton);

addHoverEffect(viewReportsButton);

addHoverEffect(removeReportsButton);

addHoverEffect(logoutButton);

}

private JButton createStyledButton(String text, Dimension size, Font font, Color backgroundColor) {

JButton button = new JButton(text);

button.setPreferredSize(size);

button.setFont(font);

button.setBackground(backgroundColor);

button.setForeground(Color.WHITE);

button.setFocusPainted(false);

button.setBorderPainted(false);

button.setOpaque(true);

return button;

}

private void addHoverEffect(JButton button) {

Color originalColor = button.getBackground();

Color darkerColor = darkenColor(originalColor);

button.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseEntered(java.awt.event.MouseEvent evt) {

button.setBackground(darkerColor);

}

public void mouseExited(java.awt.event.MouseEvent evt) {

button.setBackground(originalColor);

}

});

}

private Color darkenColor(Color color) {

float[] hsb = Color.RGBtoHSB(color.getRed(), color.getGreen(), color.getBlue(), null);

return Color.getHSBColor(hsb[0], hsb[1], Math.max(0, hsb[2] - 0.1f));

}

public static void main(String[] args) {

try {

// Set system look and feel

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (Exception e) {

e.printStackTrace();

}

SwingUtilities.invokeLater(() -> {

Dashboard dashboard = new Dashboard();

dashboard.setVisible(true);

     });

    }

}

**4.3 ADD REPORT PAGE:**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class AddReportPage extends JFrame {

private JTextField disasterNameField, disasterTypeField, locationField, severityField;

private JButton submitButton;

private JTable disasterTable;

private DefaultTableModel tableModel;

public AddReportPage() {

setTitle("Add Disaster Report");

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

setExtendedState(JFrame.MAXIMIZED\_BOTH); // Full-screen mode

// Left Panel: Form for adding disasters

JPanel formPanel = new JPanel(new GridBagLayout());

formPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(10, 10, 10, 10);

gbc.anchor = GridBagConstraints.WEST;

// Adding components to form panel

JLabel disasterNameLabel = new JLabel("Disaster Name:");

disasterNameLabel.setFont(new Font("Arial", Font.BOLD, 18));

disasterNameField = new JTextField(15);

JLabel disasterTypeLabel = new JLabel("Disaster Type:");

disasterTypeLabel.setFont(new Font("Arial", Font.BOLD, 18));

disasterTypeField = new JTextField(15);

JLabel locationLabel = new JLabel("Location:");

locationLabel.setFont(new Font("Arial", Font.BOLD, 18));

locationField = new JTextField(15);

JLabel severityLabel = new JLabel("Severity:");

severityLabel.setFont(new Font("Arial", Font.BOLD, 18));

severityField = new JTextField(15);

submitButton = new JButton("Submit Report");

submitButton.setFont(new Font("Arial", Font.BOLD, 16));

// Positioning components

gbc.gridx = 0;

gbc.gridy = 0;

formPanel.add(disasterNameLabel, gbc);

gbc.gridx = 1;

formPanel.add(disasterNameField, gbc);

gbc.gridx = 0;

gbc.gridy = 1;

formPanel.add(disasterTypeLabel, gbc);

gbc.gridx = 1;

formPanel.add(disasterTypeField, gbc);

gbc.gridx = 0;

gbc.gridy = 2;

formPanel.add(locationLabel, gbc);

gbc.gridx = 1;

formPanel.add(locationField, gbc);

gbc.gridx = 0;

gbc.gridy = 3;

formPanel.add(severityLabel, gbc);

gbc.gridx = 1;

formPanel.add(severityField, gbc);

gbc.gridx = 1;

gbc.gridy = 4;

gbc.anchor = GridBagConstraints.CENTER;

formPanel.add(submitButton, gbc);

// Right Panel: Table for displaying disasters

JPanel tablePanel = new JPanel(new BorderLayout());

JLabel tableLabel = new JLabel("Disaster Reports", JLabel.CENTER);

tableLabel.setFont(new Font("Arial", Font.BOLD, 20));

tablePanel.add(tableLabel, BorderLayout.NORTH);

tableModel = new DefaultTableModel(new String[]{"Disaster Name", "Disaster Type", "Location", "Severity", "Report Date"}, 0);

disasterTable = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(disasterTable);

tablePanel.add(scrollPane, BorderLayout.CENTER);

// Load disaster list

loadDisasterList();

// JSplitPane for dividing the window

JSplitPane splitPane = new JSplitPane(JSplitPane.HORIZONTAL\_SPLIT, formPanel, tablePanel);

splitPane.setDividerLocation(500); // Initial position of the divider

splitPane.setResizeWeight(0.5); // Equal distribution of space initially

// Add split pane to frame

add(splitPane);

// Action listener for submit button

submitButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

addDisasterReport();

}

});

}

private void addDisasterReport() {

String disasterName = disasterNameField.getText();

String disasterType = disasterTypeField.getText();

String location = locationField.getText();

String severity = severityField.getText();

// Database connection details

String url = "jdbc:mysql://localhost:3306/unisoft?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";

String user = "root"; // Change to your MySQL username

String pass = "siva2005"; // Change to your MySQL password

try (Connection con = DriverManager.getConnection(url, user, pass)) {

// Insert SQL query for disaster reports

String query = "INSERT INTO disaster\_reports (disaster\_name, disaster\_type, location, severity, report\_date) VALUES (?, ?, ?, ?, ?)";

PreparedStatement pst = con.prepareStatement(query);

pst.setString(1, disasterName);

pst.setString(2, disasterType);

pst.setString(3, location);

pst.setString(4, severity);

pst.setDate(5, new Date(System.currentTimeMillis())); // Current date for report\_date

// Execute the query

pst.executeUpdate();

// Show success message

JOptionPane.showMessageDialog(this, "Disaster report added successfully!");

// Reload disaster list

loadDisasterList();

// Clear input fields

disasterNameField.setText("");

disasterTypeField.setText("");

locationField.setText("");

severityField.setText("");

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error adding report: " + e.getMessage());

}

}

private void loadDisasterList() {

// Database connection details

String url = "jdbc:mysql://localhost:3306/unisoft?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";

String user = "root"; // Change to your MySQL username

String pass = "siva2005"; // Change to your MySQL password

try (Connection con = DriverManager.getConnection(url, user, pass)) {

// Query to fetch disaster reports

String query = "SELECT disaster\_name, disaster\_type, location, severity, report\_date FROM disaster\_reports";

PreparedStatement pst = con.prepareStatement(query);

ResultSet rs = pst.executeQuery();

// Clear existing rows in the table

tableModel.setRowCount(0);

// Populate table with data from the database

while (rs.next()) {

tableModel.addRow(new Object[]{

rs.getString("disaster\_name"),

rs.getString("disaster\_type"),

rs.getString("location"),

rs.getString("severity"),

rs.getDate("report\_date")

});

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error loading disaster list: " + e.getMessage());

}

}

// Main method for testing the AddReportPage form

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

new AddReportPage().setVisible(true); // Show AddReportPage

}

     });

    }

}

4.4 VIEW DISASTER PAGE:

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

public class ViewDisasterPage extends JFrame {

private JTable disasterTable;

private DefaultTableModel tableModel;

public ViewDisasterPage() {

// Set up the frame

setTitle("View Disaster Reports");

setExtendedState(JFrame.MAXIMIZED\_BOTH); // Full-screen mode

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

// Main panel with gradient background

JPanel mainPanel = new JPanel() {

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g;

Color color1 = new Color(240, 248, 255); // AliceBlue

Color color2 = new Color(230, 230, 250); // Lavender

GradientPaint gp = new GradientPaint(0, 0, color1, 0, getHeight(), color2);

g2d.setPaint(gp);

g2d.fillRect(0, 0, getWidth(), getHeight());

}

};

mainPanel.setLayout(new BorderLayout());

// Header label

JLabel headerLabel = new JLabel("Disaster Reports", SwingConstants.CENTER);

headerLabel.setFont(new Font("Arial", Font.BOLD, 36));

headerLabel.setBorder(BorderFactory.createEmptyBorder(20, 10, 20, 10));

mainPanel.add(headerLabel, BorderLayout.NORTH);

// Table setup

tableModel = new DefaultTableModel(new Object[]{"ID", "Disaster Name", "Type", "Location", "Severity", "Date"}, 0);

disasterTable = new JTable(tableModel);

disasterTable.setFont(new Font("Arial", Font.PLAIN, 16));

disasterTable.setRowHeight(25);

JScrollPane scrollPane = new JScrollPane(disasterTable);

// Add table to center

mainPanel.add(scrollPane, BorderLayout.CENTER);

// Load disaster data

loadDisasters();

// Add main panel to frame

add(mainPanel);

}

private void loadDisasters() {

String url = "jdbc:mysql://localhost:3306/unisoft?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";

String user = "root"; // MySQL username

String pass = "siva2005"; // MySQL password

try (Connection con = DriverManager.getConnection(url, user, pass)) {

String query = "SELECT \* FROM disaster\_reports";

PreparedStatement pst = con.prepareStatement(query);

ResultSet rs = pst.executeQuery();

// Clear any existing data

tableModel.setRowCount(0);

// Populate the table with data

while (rs.next()) {

int id = rs.getInt("id");

String disasterName = rs.getString("disaster\_name");

String disasterType = rs.getString("disaster\_type");

String location = rs.getString("location");

String severity = rs.getString("severity");

Date date = rs.getDate("report\_date");

tableModel.addRow(new Object[]{id, disasterName, disasterType, location, severity, date});

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error loading disaster reports: " + e.getMessage());

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

ViewDisasterPage viewDisasterPage = new ViewDisasterPage();

viewDisasterPage.setVisible(true);

     });

    }

}

4.5 REMOVE DISASTER PAGE:

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumn;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class RemoveDisasterPage extends JFrame {

private JTable disasterTable;

private DefaultTableModel tableModel;

private JButton removeButton;

public RemoveDisasterPage() {

setTitle("Remove Disaster Reports");

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

setExtendedState(JFrame.MAXIMIZED\_BOTH); // Full-screen mode

JPanel mainPanel = new JPanel() {

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g;

Color color1 = new Color(240, 248, 255); // AliceBlue

Color color2 = new Color(230, 230, 250); // Lavender

GradientPaint gp = new GradientPaint(0, 0, color1, 0, getHeight(), color2);

g2d.setPaint(gp);

g2d.fillRect(0, 0, getWidth(), getHeight());

}

};

mainPanel.setLayout(new BorderLayout());

// Header label

JLabel headerLabel = new JLabel("Disaster Reports", SwingConstants.CENTER);

headerLabel.setFont(new Font("Arial", Font.BOLD, 36)); // Make the header large

headerLabel.setForeground(new Color(0, 0, 128)); // Dark blue color for the header

headerLabel.setBorder(BorderFactory.createEmptyBorder(30, 10, 30, 10)); // Add padding around the header

mainPanel.add(headerLabel, BorderLayout.NORTH);

// Set up table with larger fonts and row height

tableModel = new DefaultTableModel(new Object[]{"ID", "Disaster Name", "Type", "Location", "Severity", "Date"}, 10);

disasterTable = new JTable(tableModel);

disasterTable.setFont(new Font("Arial", Font.PLAIN, 18)); // Larger font for the table content

disasterTable.setRowHeight(40); // Increase row height

// Set custom column width

TableColumn column = disasterTable.getColumnModel().getColumn(0); // ID column

column.setPreferredWidth(100); // Increase width for ID column

column = disasterTable.getColumnModel().getColumn(1); // Disaster Name column

column.setPreferredWidth(200); // Increase width for Disaster Name column

column = disasterTable.getColumnModel().getColumn(2); // Disaster Type column

column.setPreferredWidth(150); // Adjust Disaster Type column width

column = disasterTable.getColumnModel().getColumn(3); // Location column

column.setPreferredWidth(150); // Adjust Location column width

column = disasterTable.getColumnModel().getColumn(4); // Severity column

column.setPreferredWidth(100); // Adjust Severity column width

column = disasterTable.getColumnModel().getColumn(5); // Date column

column.setPreferredWidth(120); // Adjust Date column width

JScrollPane scrollPane = new JScrollPane(disasterTable);

// Set up Remove button with larger size and increased font

removeButton = new JButton("Remove Selected Disaster");

removeButton.setFont(new Font("Arial", Font.BOLD, 24)); // Larger font for the button

removeButton.setPreferredSize(new Dimension(400, 60)); // Larger button

removeButton.setEnabled(false); // Initially disabled until a row is selected

// Enable button when a row is selected

disasterTable.getSelectionModel().addListSelectionListener(e -> {

removeButton.setEnabled(disasterTable.getSelectedRow() != -1);

});

// Button panel

JPanel buttonPanel = new JPanel();

buttonPanel.add(removeButton);

// Add components to frame

setLayout(new BorderLayout());

add(mainPanel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

add(buttonPanel, BorderLayout.SOUTH);

// Load data from the database

loadReports();

// Action listener for the remove button

removeButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

removeSelectedDisaster();

}

});

}

private void loadReports() {

String url = "jdbc:mysql://localhost:3306/unisoft?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";

String user = "root"; // MySQL username

String pass = "siva2005"; // MySQL password

try (Connection con = DriverManager.getConnection(url, user, pass)) {

String query = "SELECT \* FROM disaster\_reports"; // Query to fetch all reports

PreparedStatement pst = con.prepareStatement(query);

ResultSet rs = pst.executeQuery();

// Clear previous data in the table

tableModel.setRowCount(0);

// Iterate through the result set and add each row to the table

while (rs.next()) {

int id = rs.getInt("id");

String disasterName = rs.getString("disaster\_name");

String disasterType = rs.getString("disaster\_type");

String location = rs.getString("location");

String severity = rs.getString("severity");

Date reportDate = rs.getDate("report\_date");

// Add row to table model

tableModel.addRow(new Object[]{id, disasterName, disasterType, location, severity, reportDate});

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error loading reports: " + e.getMessage());

}

}

private void removeSelectedDisaster() {

int selectedRow = disasterTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "No disaster selected!");

return;

}

int id = (int) tableModel.getValueAt(selectedRow, 0); // Get ID from the selected row

String url = "jdbc:mysql://localhost:3306/unisoft?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";

String user = "root"; // MySQL username

String pass = "siva2005"; // MySQL password

try (Connection con = DriverManager.getConnection(url, user, pass)) {

String query = "DELETE FROM disaster\_reports WHERE id = ?"; // Query to delete the selected report

PreparedStatement pst = con.prepareStatement(query);

pst.setInt(1, id);

// Execute deletion

int rowsAffected = pst.executeUpdate();

if (rowsAffected > 0) {

// Remove the row from the table model

tableModel.removeRow(selectedRow);

JOptionPane.showMessageDialog(this, "Disaster report removed successfully!");

} else {

JOptionPane.showMessageDialog(this, "Failed to remove the disaster report!");

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error removing report: " + e.getMessage());

}

}

// Main method for testing the RemoveDisasterPage form

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

new RemoveDisasterPage().setVisible(true); // Show RemoveDisasterPage

}

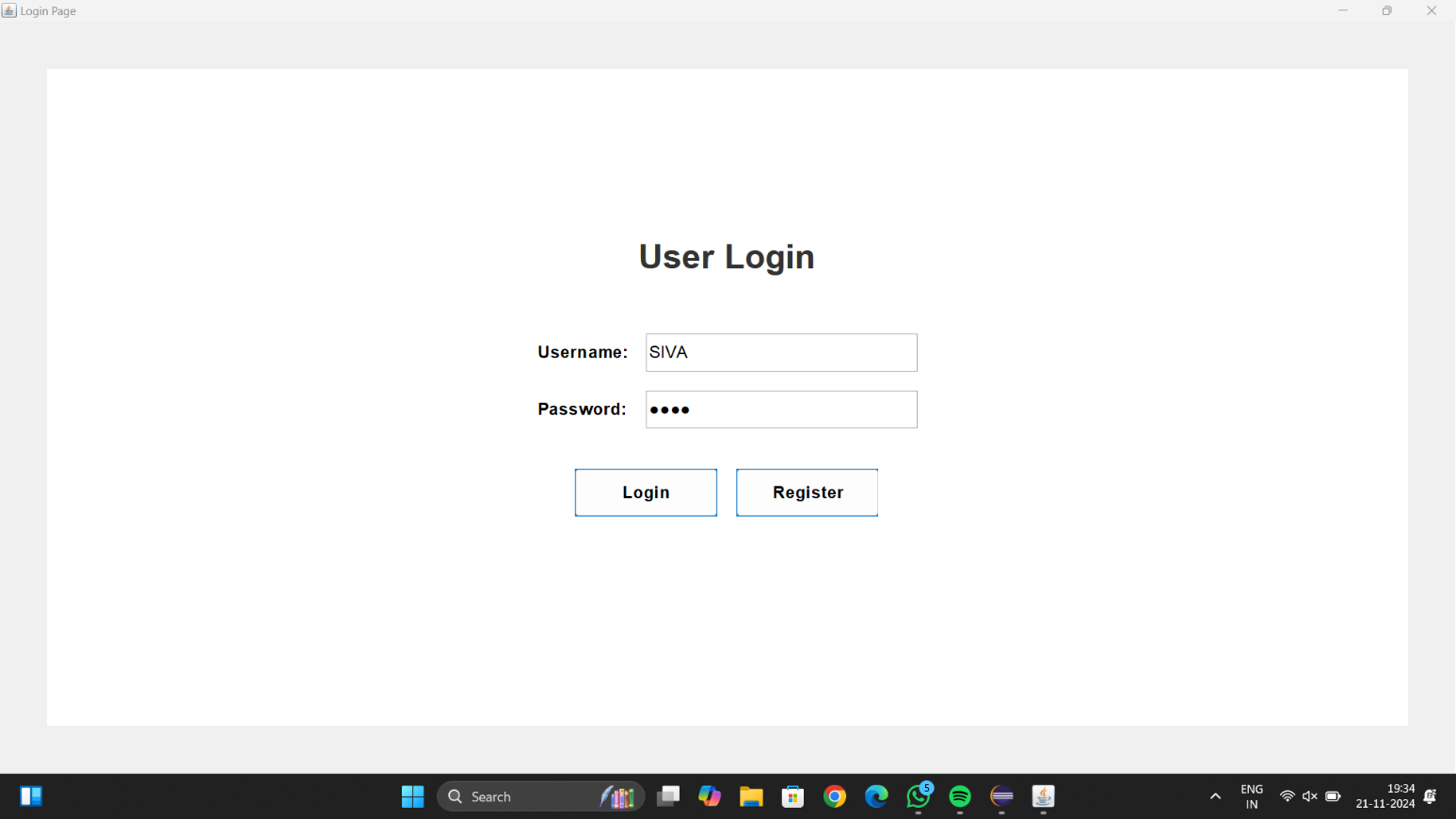
     });

    }

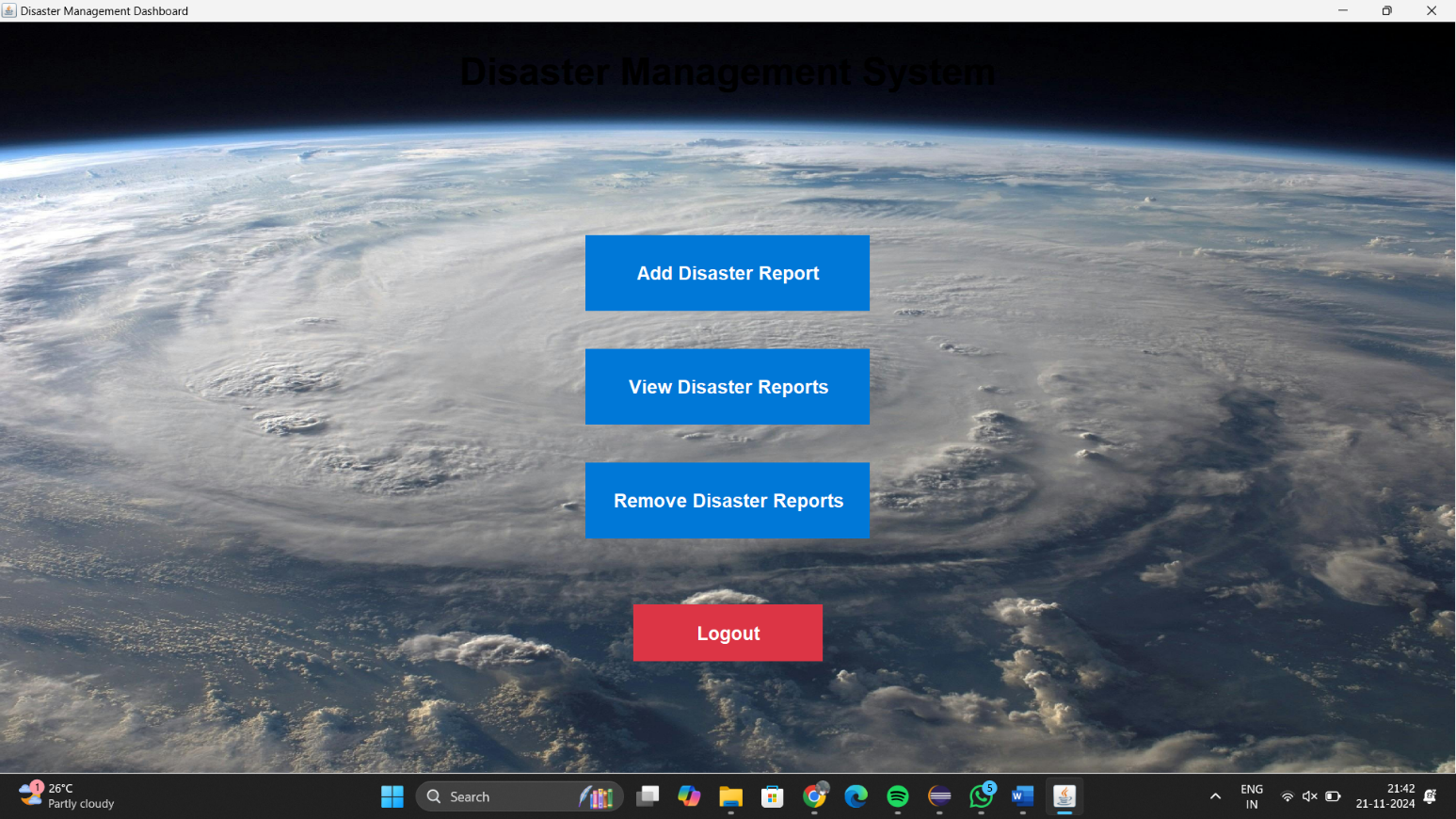
}

## SNAPSHOTS

**5.1 LOGIN PAGE:**

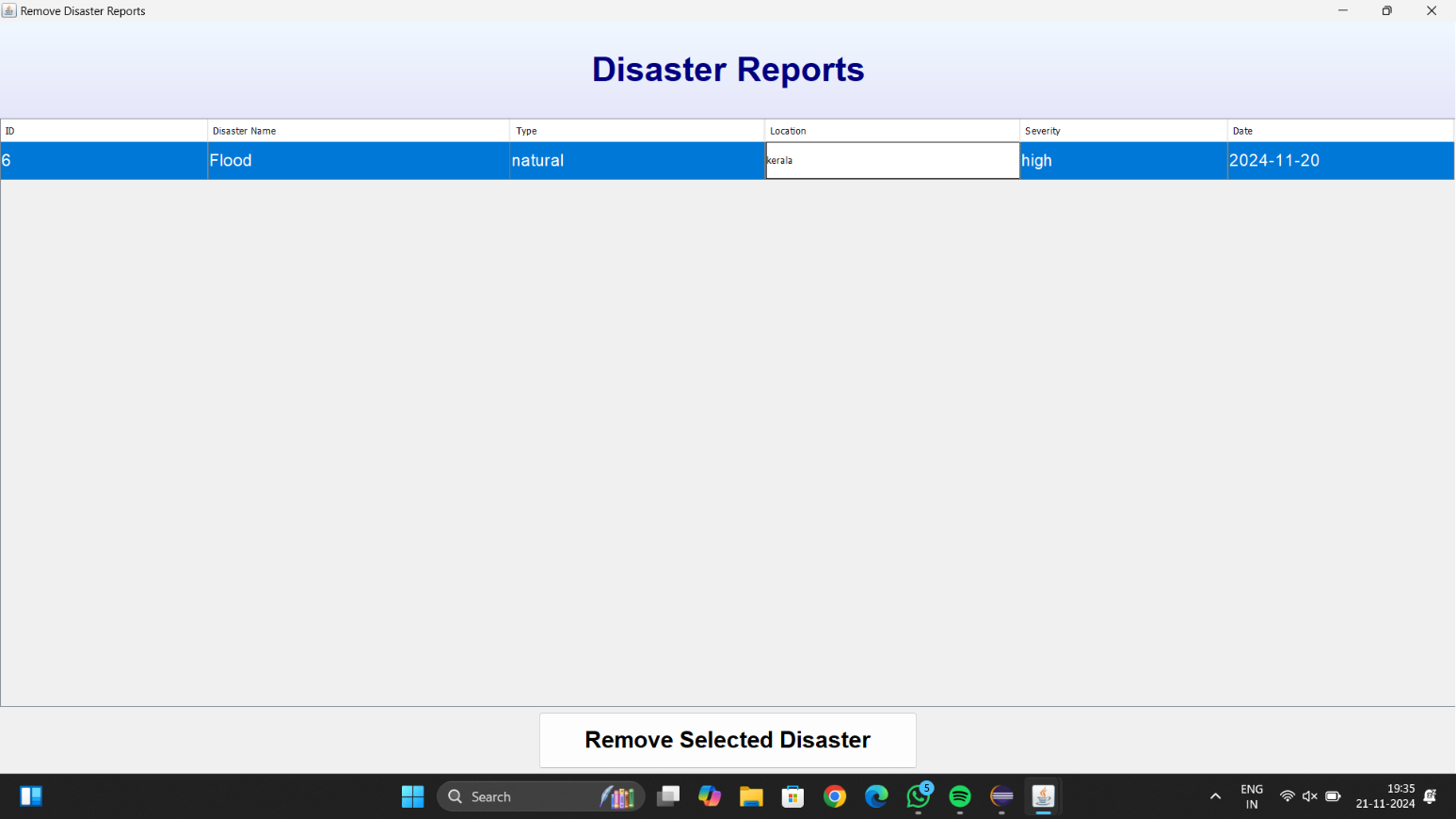


### 5.2 DASHBOARD



### 5.3 ADD REPORT PAGE

### 5.4 REMOVE DISASTER PAGE



### 5.5 VIEW DISASTER PAGE

## CONCLUSION

In conclusion, a **Disaster Management System** is a critical tool for mitigating the impact of disasters by enabling efficient preparation, swift response, and organized recovery efforts. By leveraging technology for early warnings, resource management, and coordination, it ensures better protection of lives, property, and the environment. The system not only addresses immediate disaster needs but also helps in building long-term resilience and improving preparedness for future crises. Ultimately, it fosters a more proactive and effective approach to disaster management, benefiting communities and organizations alike.

## REFERENCES

1. [*https://www.javatpoint.com/java-tutorial*](https://www.javatpoint.com/java-tutorial)
2. *https://www.wikipedia.org/*
3. *https://www.w3schools.com/sql/*